Use of the Novus Bluetooth Serial Library for myMIP

James Strawson 12/3/12

Part 1: Basics

Connect to your arduino nano as follows: **vcc > 5v, gnd > gnd, rx > tx, tx > rx.**

The intent of this library is to allow you to type various instructions into a serial monitor and easily interpret these instructions in your arduino code. The main functions you will call are:

Bluetooth.refresh()

This asks the arduino to parse any new incoming data and update its memory with new values. This function returns a Boolean (true or false) to indicate if anything is new. Thus, it can be placed in an if-statement that only executes code when new values appear. You should call this at regular intervals (50hz for example). Polling much faster than this may result in erratic behavior and is not necessary.

```
if (Bluetooth.refresh()){
    //your code here
}
```

Bluetooth.getForwardSpeed() & Bluetooth.getTurnRate()

These two functions return a signed floating point value containing whatever the most recent command sent from your computer/phone.

Bluetooth.getAux1(), Bluetooth.getAux2() & Bluetooth.getAux3()

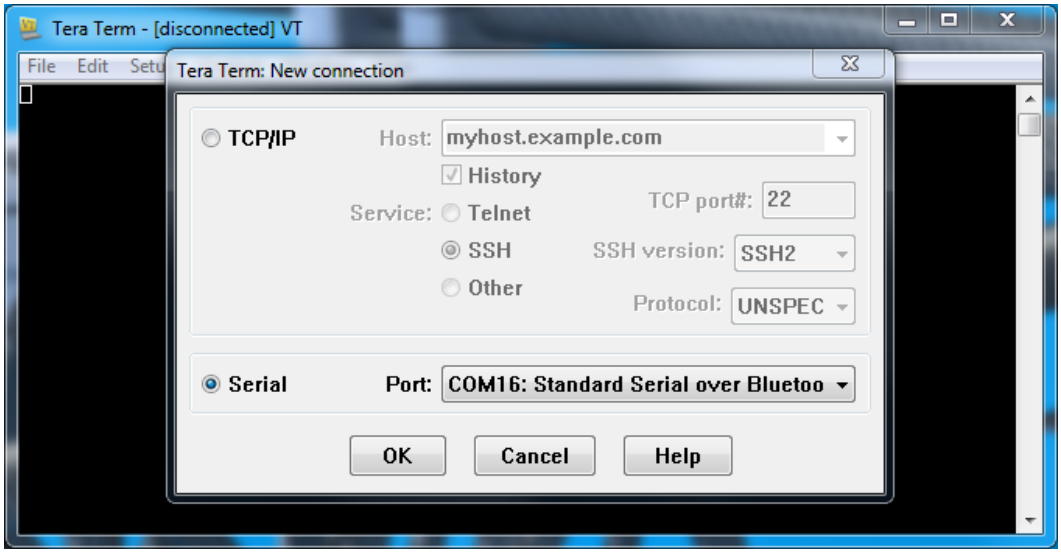
These three auxiliary commands give you three more channels through which to send data. These could be used for real-time updating of controller parameters or perhaps to tell your myMip to do a little dance. They return floating point values the same as Forward Speed and TurnRate.

How do I send & receive commands from my phone or laptop? BlueTerm is my preferred android terminal and is free on the android market. For me, getting blueterm working only requires pairing to the device in Bluetooth settings followed by opening blueterm and selecting which device to connect to in the menu.

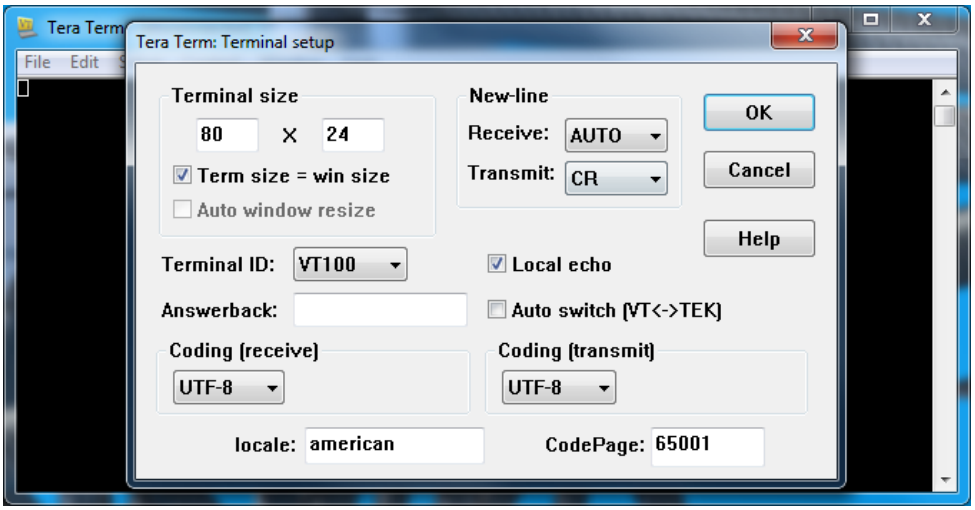
TeraTerm is my preferred PC terminal and can be found here:

http://en.sourceforge.jp/projects/ttssh2/downloads/57572/teraterm-4.76.exe/

After pairing your laptop and Bluetooth device using your operating system's Bluetooth device manager and the passkey "1234," opening teraterm should present you with two options in the serial dropdown menu as shown below. Typically the lower number is the working port. Try both if you are having connectivity problems.



In settings>terminal, you should change the “newline, receive” option to auto since arduino uses newline instead of carriage return line ending. Enabling “local echo” will allow you to see what you are typing on the screen.



A solid red light on your Bluetooth module indicates it is connected to either teraterm or blueterm.

From there, you specify the turn rate by typing a letter ‘t’ or ‘T’ immediately followed by any number you want to send to the arduino, then hit enter to send the data. Negatives and decimals are allowed. Any other characters are ignored. All of the following set the output of Bluetooth.getTurnRate().

T123
t-1.23
oefwhgpoiT1.23wpj;rg{wg

To set the forward speed use the indicator ‘f’ or ‘F’ instead, OR, if you type nothing but a number, it will assume you mean forward speed. All of the following set the forward speed value.

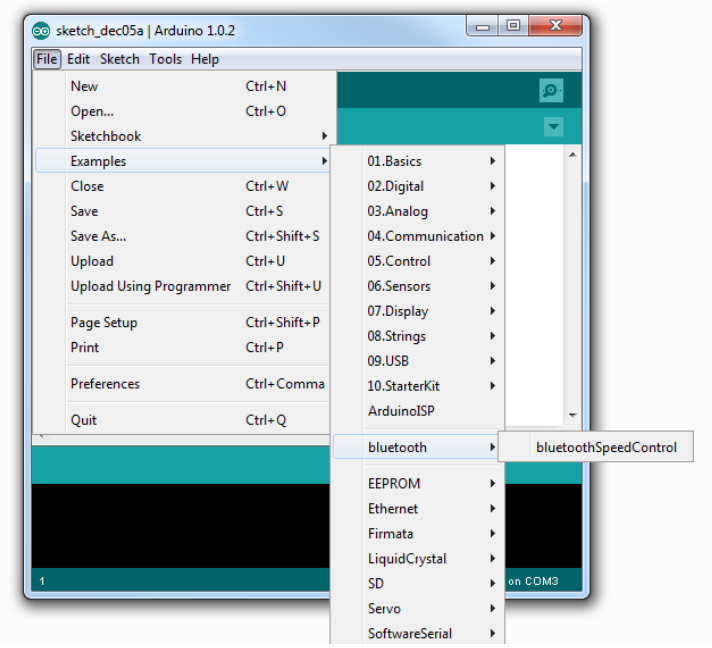
,F-3.14,
f3.14
-3.14

The three Auxiliary channels are indicated with upper or lowercase A, B, & C respectively. Multiple commands can be stringed together; the following lines all set the forward speed to 3.14, the turn rate to 1.23, and the Aux2 channel to 1003 in a single call of Bluetooth.refresh() and a single command from your serial monitor.

T1.23, 3.14, b1003
B1003 kjhk T1.23][{} f3.14
T1.23 3.14 b1003
You will probably use this format most frequently to set just forward speed and turn rate: ***1.23t3.14***

Each line you send should be less than 64 characters long, and each number should be less than 16 characters long including the decimal and minus sign if you use them. This gives you the freedom to decide how your arduino interprets what you send it. For example, you may want 100 to be full forward, and -100 to be full back, or perhaps 1 & -1 instead.

You should place the entire “bluetooth” folder in the libraries directory of your “arduino-1.0x” folder which you downloaded at the beginning of the class. If you moved your sketchbook directory then you may create a “libraries” folder in your sketchbook directory and use that for convenience. This is the entire installation procedure. From here, you should be able to open the demo program I showed in class either by opening the .ino file in the examples folder or straight from your arduino environment as shown below.



Here is a sample arduino program that just sends back the new state of forwardSpeed and turnRate whenever new values are sent through Bluetooth or USB. This is the typical way of polling non-system-critical functions at regular intervals without interrupts and is here for you to reference when you write more arduino programs in the future. This document should stay in your libraries directory for reference too.

```
#include <bluetooth.h>                  //Make sure bluetooth.h and Bluetooth.cpp are in your arduino library folder
bluetooth Bluetooth;                    //Creates a new Instance of the bluetooth class
long lastLoopMillis;                    //used for keeping track of the main loop execution interval

void setup(){
 Serial.begin(115200);                  //must always start serial at beginning of your code
}

void loop(){

 if(millis() - lastLoopMillis >= 20){    //makes sure 20ms has passed since last call (50hz)
  if (Bluetooth.refresh()){             //the refresh function returns true if there is new information
   Serial.print(Bluetooth.getForwardSpeed());   //Print new information
   Serial.print("\t");
   Serial.println(Bluetooth.getTurnRate());
  }
  lastLoopMillis = millis();            //reset the timer
 }
}
```

Part 2: Advanced

So now you want to be fancy eh? You may have already noticed that the Bluetooth class stores the data being sent through each channel as a floating point number in an array. There are 32 addresses in this array assigned as follows

Address	0	1	2	3-5	6-31
Assignment	reserved	forwardSpeed	turnRate	Aux1 – Aux3	Free

To assign a value to any one of the addresses from your serial monitor or phone simply replace the letters used in the basic section with the address followed by a semicolon and the new value. For example to set forwardSpeed to 3.14 and address 6 to 9001, you could type any of the following.

3.14 6:9001
1:3.14,6:9001
6:9001 wjrhtg [] f3.14

Your arduino can read any of the floating point values in this array by handing the following function the address you would like to read.

float getRegister(int address);

If you would like to manually set the value of a register stored in the Bluetooth class memory, call the function.

void setRegister(int address, float value);